目次

改訂情報

変更年月日	変更内容
2012-12-21	初版
2013-12-20	第2版 下記を追加・変更しました ▪ ドキュメント全般 Windows Server 2012 向けの記述を追加
2014-07-04	第3版 下記を追加・変更しました ▪ プログラミングガイドの内容をセットアップガイドに合わせました。
2014-12-01	第4版 下記を追加・変更しました ▪ ドキュメント全般 Windows Server 2012 R2 向けの記述を追加 ▪ IOCELA（連票形式）の出力カスタマイズの項目を追加 ▪ その他（便利な機能）の記述を追加
2015-07-01	第5版 下記を追加・変更しました ▪ iAPへのバージョンアップ時の注意事項の記述を追加 ▪ その他（便利な機能）の記述を追加

はじめに

本書の目的

本書ではIM-PDFDesigner for Accel Platform を利用したユーザプログラムを開発する場合の基本的な方法や注意点等について説明します。IM-PDFDesigner for Accel Platform を利用して開発を行う前にお読み下さい。

対象読者

本書は、開発をスムースに開始するための手引書となっています。
したがって、実際に開発を行うプログラマの方が対象となります。

また、本書は、以下に列挙する技術に関する知識を有することを前提として構成されています。
これらの技術に関して不明な点がある場合、本ドキュメントの内容を正しく理解することが困難 になることがありますので、予めご了承ください。なお、前提知識となる技術に関しては、一般 の専門書籍等をご覧ください。

- Javaプログラミング言語
- Java ServletおよびJSP
- オペレーティングシステム
- ネットワーク

用語解説

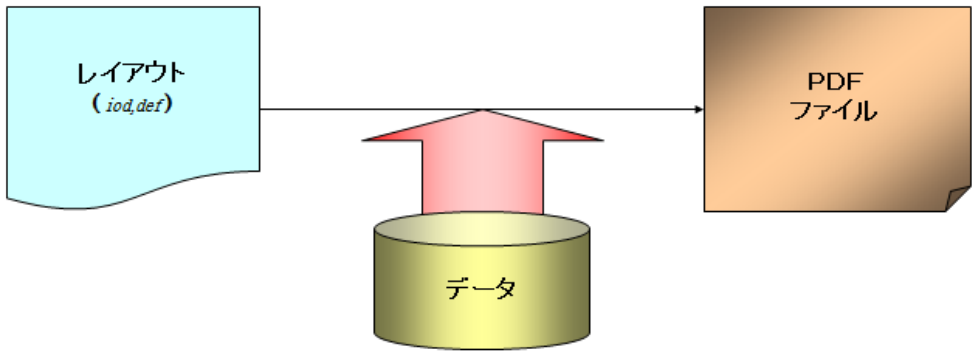
intra-mart Accel Platform	iAP と略します。
IM-PDFDesigner for Accel Platform	PDFデザイナー と略します。
iAP ホームディレクトリ	%HOME_PATH% と略します。
Storageのディレクトリ	%PUBLIC_STORAGE_PATH% と略します。
帳票エンジンのディレクトリ	%IOWEBDOC_PATH% と略します。

概要

PDFデザイナー を利用したプログラム開発は以下の流れになります。

1. レイアウト定義ファイルの作成
2. PDFファイル作成用のプログラム開発

> **コラム**
>
> PDFファイル作成の基本的な概念は、下図のようにレイアウト定義に対してデータを埋め込んでPDFファイルを作成します。
>
> つまり、1つのレイアウトから1つのPDFファイルが作成されることが基本となります



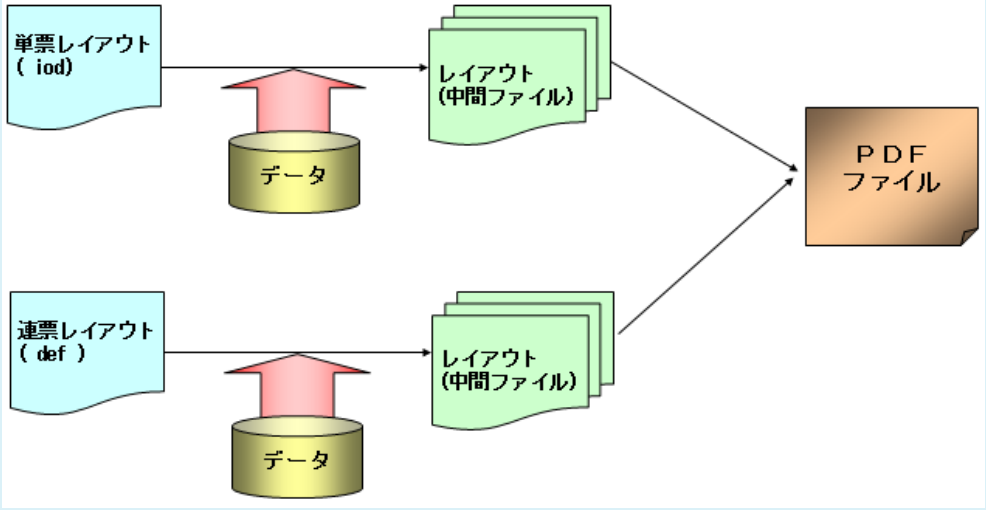
.

データは、文字列データを指定する以外に制約はありません。従って、データベースから取得したデータをはじめとして、様々なデータソースから取得した値を指定することができます。

データを指定方法としては、以下3種類の方法がご利用いただけます。

- CSVファイルを指定する。
- DATファイルを指定する。
- プログラム中でデータ文字列を動的に指定する。

> **コラム**
>
> PDFファイルの作成方法としては、複数のレイアウトから1つのPDFファイルを作成する方法も用意されています。
>
> この方法で、形式のことなる帳票レイアウトを1つのPDFファイルに束ねることが可能になります。



上記のような方法を応用することにより、複数のレイアウトを組み合わせて複雑なドキュメントを作成することも可能です。

コラム

本製品を利用して作成したPDFファイルの表現技法を良く理解した上で、どのようなドキュメントを作成したいのかを設計段階で十分に検討をすることが非常に重要になります

レイアウト定義ファイルの作成

PDFファイルのレイアウトを設計したら、レイアウトデザインツールを利用して レイアウト定義ファイルを作成します。

レイアウトデザインツールは、Windows環境で動作します。UNIX/Linux環境を利用 される場合でも、レイアウト定義はWindows環境でおこないます。

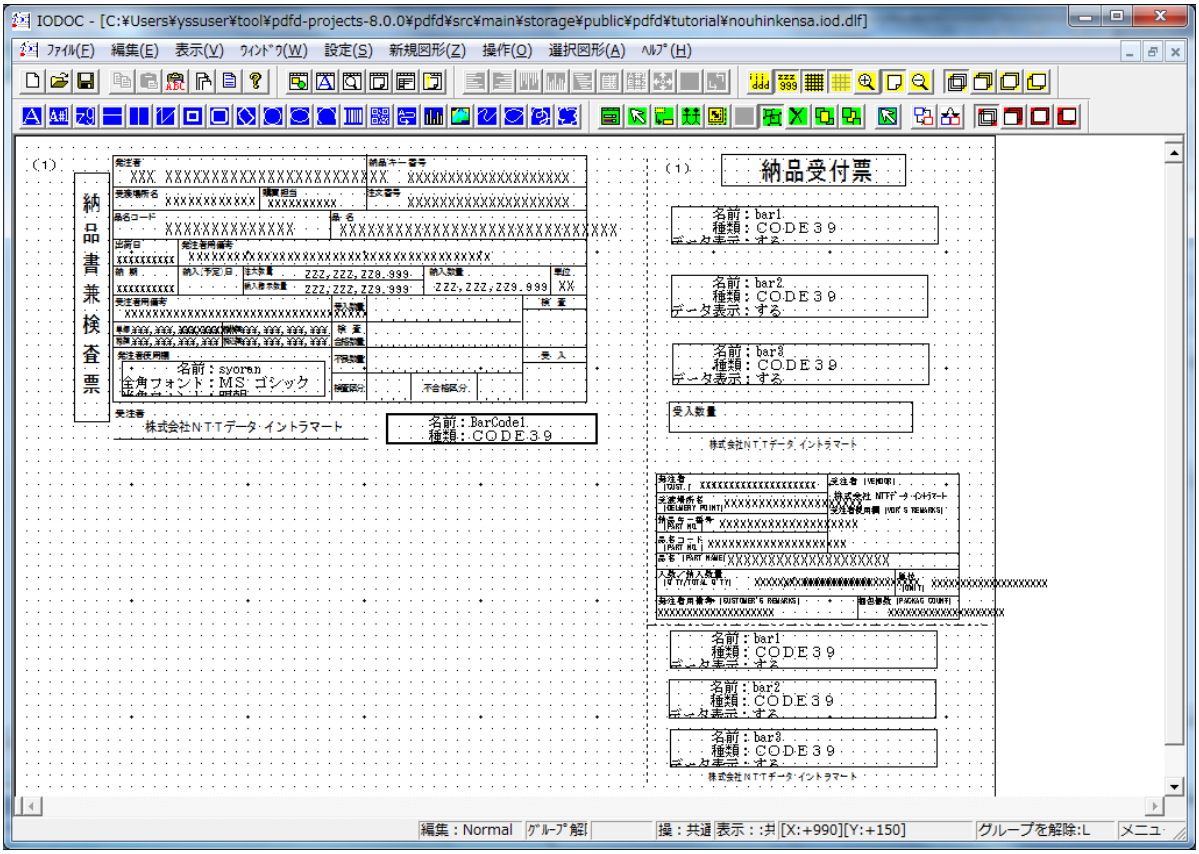
利用フォントは、等幅フォントとしてください。可変幅フォントは利用できません。
（例）MSゴシック→OK、MS Pゴシック→NG

単票用 (IODOC)

単票用(IODOC)は、複雑なレイアウトのページの作成に適しています。

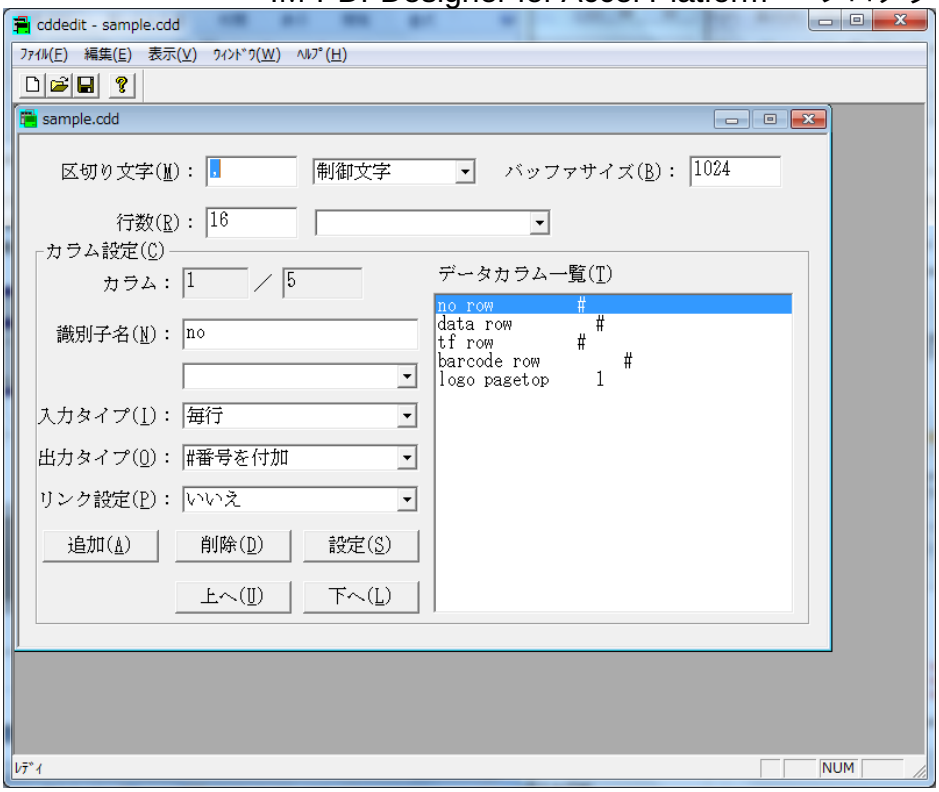
[スタート]-[すべてのプログラム]-[IOWebDOC Vx.x.x.x]-[IODOC]を実行すると、単票用レイアウト定義ファイルを作成するためのツールが起動します。（x.x.x.xの部分にはバージョン番号が入ります）

IODOCの使い方に関しては、専用のマニュアル tool/document/iodoc_Tool.pdf をご覧ください。



また、PDFファイルを作成する時にCSV形式のデータと連携させる場合、変換定義ファイル（cdd）が必要になります。この変換定義ファイルは、CDDエディタを利用すると簡単に作成することができます。
CDDエディタの使い方に関しては、専用のマニュアル tool/document/cddedit.pdf をご覧ください。

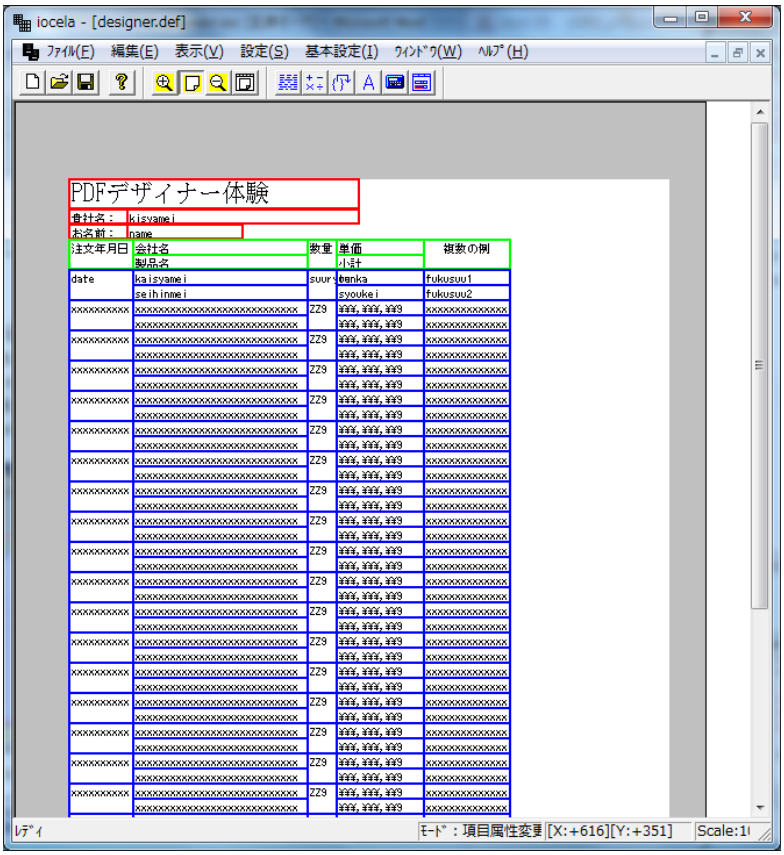
[スタート]-[すべてのプログラム]-[IOWebDOC V1.9.2.3]-[CDDエディタ]を実行すると、CDDエディタ が起動します。



連票用 (IOCELA)

連票用(IOCELA)は、連続した表形式の帳票の作成に適しています。

[スタート]-[プログラム]-[IOWebDOC V1.9.2.3]-[IOCELA]を実行すると、連票用レイアウト定義ファイルを作成するためのツールが起動します。
IOCela の使い方に関しては、専用のマニュアル tool/document/iocela_Tool.pdf をご覧ください。

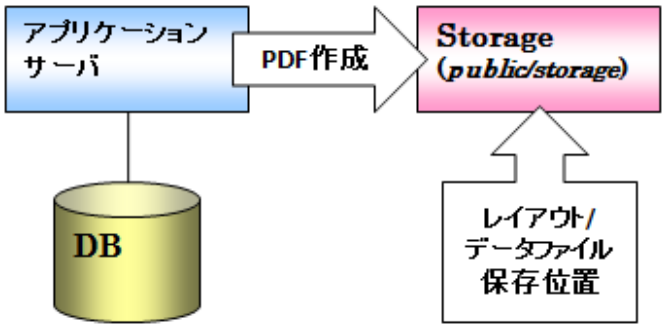


PDFファイル作成用のプログラムの開発

本製品で提供されているAPIを用いて、PDFファイルを作成するためのプログラムを作成します。
PDFファイルを作成するためのAPIの利用方法については、PDFデザイナーをインストールすると intra-martサーバに自動的にプログラムがインストールされ各開発モデル用のAPIをご利用頂けます。各APIの仕様に関しては、APIリストが提供されていますので、そちらをご利用ください。
サンプルプログラムが用意されていますので、API仕様を学ぶ場合にご利用ください。

動作概念

PDFデザイナー では、PDFファイルの作成に必要な各ファイル (レイアウトなど) は、
%PUBLIC_STORAGE_PATH%/public/storage 以下の任意のフォルダに保存しておく必要があります。
また、作成されるPDFファイルも同じく%PUBLIC_STORAGE_PATH%/public/storage 以下の指定のフォルダに 出力されます。



> コラム
>
> PDFデザイナーの各レイアウト/データファイル及びPDFファイル出力位置は、
> %PUBLIC_STORAGE_PATH% を C:/tmp/storage とした場合
>
> C:/tmp/storage/public/storage/以下の任意のフォルダ
>
> となります。
>
> PDFデザイナーでは、各ファイルの位置は上記のフォルダからの相対パスで指定します。

スクリプト開発モデル

本製品によって iAP に追加されたPDF作成用のAPIは、iAP が標準で提供している他のAPIと同様に利用することができます。

JavaEE開発モデル

プログラムのコンパイル

PDFデザイナーのAPIを利用したプログラムをコンパイルする場合、IM-PDFDesigner for Accel Plotform / セットアップガイドで指定した jarファイル をクラスパスに設定して下さい。

プログラム開発における注意点

- 作成した PDFファイル のファイルサイズが大きい場合、APIのレスポンスとPDFファイルがディスク上に完全に書き出されるタイミングが大きく異なる場合があります。サイズの大きいPDFファイルを作成した場合は、十分な時間が経過した後に作成したPDFファイルにアクセスするようにして下さい。
- CSVファイルに画像のパスを指定する場合は、絶対パス　もしくは　カレントディレクトリからの相対パス　にて指定ください。

- PDFデザイナーが提供するAPIは、指定されたパスを Public Storage (標準では、%PUBLIC_STORAGE_PATH%/public/storage/) を親ディレクトリとして解決します。
  したがって、上記パスからの相対パスを指定して下さい。
- PDFデザイナーが提供するAPIでは、Public Storage 以下のすべてのファイルに対してアクセスすることができますので、PDFファイル作成により不用意にファイルを上書きしてしまわないように注意して下さい (指定されたパスが、すでにファイルとして存在していても、API実行の結果エラーになることはありません)。
- 例外として、CSVファイルでデータを渡す場合は、記載するファイルのパス（画像ファイル等…）はOSの認識するフルパスにて指定ください。
  %PUBLIC_STORAGE_PATH% からの相対パスでは指定できませんのでご注意ください。
- レイアウト定義ファイルに関しては、通常は Shift-JIS で作成されます。
- PDFデザイナーにおいて、通常のJavaアプリケーション同様にファイル出力が競合しないよう、上位アプリケーション側でファイルの排他制御が必要になります。
  - システムで重複しない出力ファイル名を使用する。
  - ダブルクリックを防止する

  出力ファイル名については、上位アプリケーション側でシステムで重複しない IDなどを生成し、ファイル名に付加してご利用下さい。
  ダブルクリック防止機能については、スクリプト開発モデルであれば isDoubleClick() がご利用頂けます。
  JavaEE開発モデルであれば DblClickForbidenタグ がご利用頂けます。ともにイントラマートに標準のAPIとなっています。詳細については、イントラマートAPIマニュアルを ご参照下さい。
- PDFデザイナーは、PDFファイルに2種類のパスワードを設定することが可能です。
  - オープンパスワード
    PDFファイルの閲覧を制限するためのパスワードです。AdobeReader等で開く際にパスワードが要求されます。
  - セキュリティパスワード
    PDFファイルに対して、編集・加工等の操作を制限するためのパスワードです。Adobe Acrobat等で編集する際にパスワードが要求されます。

チュートリアル

ここでは、本製品のAPIを利用して、スクリプト開発モデル・JavaEE開発モデルについて、実際にプログラムを作成する過程を説明します。
チュートリアルでは、説明を簡素化するためJSPからファイルダウンロードを行っていますが、実運用ではサーブレットを使用してファイルをダウンロードして下さい。

前提条件

- iAP が正しくインストールされていて、正常に動作していること。
- 各アプリケーションサーバに PDFデザイナーが正しくインストールされていてること。

準備

- レイアウトファイルを %PUBLIC_STORAGE_PATH%/public/storage の任意の位置に保存して下さい。

プログラム開発

スクリプト開発モデル

単票形式、連表形式について、実際に帳票を出力するまでの過程を説明します。

単票形式

項目

単票形式のPDFファイルを作成するためのスクリプトプログラムを作成します。
スクリプト開発では、htmlファイルとJavaScriptファイルを作成する必要があります。

このチュートリアルでは、サンプルとしてインストールされているレイアウトファイルを利用しています。

コラム

文字コードは UTF-8 でファイルを保存して下さい。

1. 入力画面の作成

テキストエディタを起動して、以下のHTMLを記述します。

```
<!--
// CSVDOCサンプル(PDF-Desiner V8.0.0)
// IODoc帳票データをコード内部で生成し、単票用レイアウトの
// PDF帳票ファイルを生成します。
// PDFファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
   -->
<DIV align="center" style="center; padding-top: 25px;">
  <P><FONT size="+2">チュートリアルサンプル(IODoc)</FONT></P>
  <TABLE border="1">
    <TR>
      <TH align="center" style="padding: 5px 10px;" nowrap>
        出力PDFファイルは、<BR>Public Storageの[pdfd/tutorial]<BR>
        フォルダ下に作成され、処理終了後に自動ダウンロードされます。
      </TH>
    </TR>
    <TR>
      <TH align="center" style="padding: 5px 10px;" nowrap>
        処理実行は下のボタンをクリックします。
      </TH>
    </TR>
    <TR>
      <TD align="center" style="padding: 5px 10px;" nowrap>
        <IMART type="form" action="makePDF">
          <INPUT type="submit" value=" PDF作成 ">
        </IMART>
      </TD>
    </TR>
  </TABLE>
</DIV>
```

記述が完了したら %HOME_PATH%/jssp/src/pdfd/tutorial ディレクトリを作成し、
docsample.html というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

2. 入力画面処理の作成

次にJavaScriptファイルを作成します。
“//”から始まる行は、コメントですので無視して記述頂いても問題ありません。

```
//---
// CSVDOCサンプル(PDF-Desiner V8.0.0)
//---
// IODoc帳票データをコード内部で生成し、単票用レイアウトのPDF帳票ファイルを生成します。
// PDFファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
function makePDF(request) {

    //-----
    // 出力ファイルパスの設定
    //-----
    // 出力ファイルは、Storage上のPublicディレクトリ以下の任意の位置に出力できます。
    //
    //    Publicディレクトリとは、
    //        %PUBLIC_STORAGE_PATH%/public/storage/ までを指しています。
    //
    // また、ファイル名は PublicStorageクラスを使用しStorage上に同一ファイルがないか確認し
    // 同一ファイルが存在する場合は、ファイル名に"_"(アンダーバー)+数値"を付加しています。
    //
    // *** このサンプルでは完全な一意性は確保できません。***
    //
    var sessionid  = Client.identifier();   // セッションIDの取得
    var dirPath    = "pdfd/tutorial/";      // 出力フォルダ
    var prefix     = "nouhinkensa";         // 出力ファイル接頭文字
    var suffix     = ".pdf";                // 出力ファイル拡張子
    var outPdfName = prefix + "_" + sessionid + suffix;
    var outPdfPath = dirPath + outPdfName;

    var ps = new PublicStorage(outPdfPath);
    var i = 1;
    while (ps.exists()) {
        outPdfName = prefix + "_" + sessionid + "_" + i + suffix;
        outPdfPath = dirPath + outPdfName;
        ps = new PublicStorage(outPdfPath);
        i++;
    }

    //-----
    // インスタンス生成 (V8.0.0から変更あり)
    //-----
    // メモリオブジェクト方式のため、入力IODのみ指定。
    // CSVファイル形式の場合はCCDファイルも指定します。
    //
    // 【V8.0.0】ファイルパス指定方法の変更
    //         Storage の Publicディレクトリ からの相対パスを指定します。
    //
    var pdf = new IODoc("pdfd/tutorial/nouhinkensa.iod", "");

    //-----
    // 文書情報設定 (V7.x から変更なし)
    //-----
    // 文書情報セット（各項目最大255文字まで）
    //
    // defineTitle(String)       タイトルの設定
    // defineSubTitle(String)    サブタイトルの設定
    // defineAuthor(String)      作成者の設定
    // defineApplication(String) 作成アプリケーション名の設定
```

```
57      //
58      pdf.defineTitle("PDFデザイナー体験");
59      pdf.defineAuthor("IM 太郎");
60
61      //-----
62      // セキュリティ設定 (V7.x から変更なし)
63      //-----
64      // セキュリティ情報セット（パスワードは最大32文字まで）
65      //
66      // <パスワード設定>
67      //    setOpenPassword(String)     オープンパスワード(32文字まで)
68      //    setSecurityPassword(String) セキュリティパスワード(32文字まで)
69      //
70      // <印刷許可設定>
71      //    printSecurity("PRINT_ENABLE")  印刷許可
72      //    printSecurity("PRINT_DISABLE") 印刷不許可
73      //
74      // <変更許可設定>
75      //    modifySecurity("MODIFY_DISABLE")            変更不許可
76      //    modifySecurity("MODIFY_ALL")                変更許可
77      //                                    (ページの抽出を除くすべての変更を許可)
78      //    modifySecurity("MODIFY_FORM_AND_ANNOTATION") 変更許可
79      //                                    ("注釈の作成","フォームフィールドの入力",
80      //                                     "既存の署名フィールドに署名"を許可)
81      //    modifySecurity("MODIFY_FORM_AND_ASSEMBLY")   変更許可
82      //                                    ("ページレイアウト",
83      //                                     "フォームフィールドの入力",
84      //                                     "既存の署名フィールドに署名"を許可)
85      //
86      // <テキスト文字抽出許可及びアクセシビリティ許可設定>
87      //    copySecurity("COPY_AND_ACCESSBILITY_DISABLE") 不許可
88      //    copySecurity("COPY_AND_ACCESSBILITY_ENABLE")  許可
89      //
90      pdf.setSecurityPassword("secpasswd");
91      pdf.printSecurity("PRINT_DISABLE");
92      pdf.modifySecurity("MODIFY_DISABLE");
93      pdf.copySecurity("COPY_AND_ACCESSBILITY_DISABLE");
94
95      //-----
96      // ページデータの生成 (V7.x から変更なし)
97      //-----
98      // 本チュートリアルでは、メモリオブジェクト形式でのデータ設定を実施します。
99      // CSV/DATファイルオブジェクトでデータを与える場合には、
100     // ここでそれぞれ入力ファイルを設定します。
101     //
102     // 埋め込み識別子及びデータ
103     // (DBデータ検索等により取得、又はコード内で埋め込みデータを生成することの可能)
104     //
105     var doc_data = new Array(2);
106     doc_data[0] = new Array(
107           "kyakusaki","OrderComNo","nouhin_No","tantou","nouhinsaki",
108           "tyuumon_No","hinmei_code","hinmei","h_memo","syukka_day",
109           "suuryou","tani","tanka","j_memo","nouki",
110           "shiji_suuryou","nounyu_suuryou","konpou_suuryou","zei","zeinuki",
111           "zeikomi","BarCode1","bar1","bar2","bar3");
112     doc_data[1] = new Array(
113           "NTTデータイントラマート","001","001-001","IM 太郎","IM商事",
```

```
            "C-001-001","YPDFAUTO-001","IM-PDFオートコンバータ","","2004/07/01",
            "2","式","1000000","","2008/07/05",
            "2","2","2","100000","2000000",
            "2100000","CODE39","CODE39","CODE39","CODE39");

    //-----
    // テキスト関連識別子データ埋め込み (V7.x から変更なし)
    //-----
    // 通常テキスト文字列を埋め込みします。
    // (※複数行カラムデータの識別子名へは、[識別子#行番号]と編集してセットします)
    //
    for(var i = 0;i < doc_data[0].length;i++) {
        pdf.setData(doc_data[0][i],doc_data[1][i]);
    }

    //-----
    // 文字枠データの埋め込み (V7.x から変更なし)
    //-----
    // 文字枠への出力は下記の順で実施する。
    //    開始宣言[setTextBoxStart]→データセット[setTextBoxData]→終了宣言[setTextBoxEnd]
    //
    // データセットは1回で1行分のデータを出力できます。
    // (文字枠より大きい文字列長のデータが指定された場合は自動改行されます。)
    // 複数回データセットを呼び出すことで、改行を含めた文字列のセットが可能です。
    //
    pdf.setTextBoxStart("syoran");
    pdf.setTextBoxData("至急納品");
    pdf.setTextBoxEnd();

    //-----
    // ページ区切りを出力 (V7.x から変更なし)
    //-----
    // 複数ページとなる伝票を印刷する場合、印刷ページ区切りを指定することで改ページ位置を指定できます。
    //
//  pdf.setOutPage();

    //-----
    // PDF出力処理 (V8.0.0 から変更あり)
    //-----
    // PDFファイルへの出力処理が実行されます。
    // 正常に処理が完了した場合には指定されたPDFファイル名に該当の文書が作成されます。
    //
    // 【V8.0.0】ファイルパス指定方法の変更
    //         Storage の Publicディレクトリ からの相対パスを指定します。
    //
    var resultCode = pdf.toPDF(outPdfPath);

    //-----
    // 終了処理 (V8.0.0 から変更あり)
    //-----
    // PDF作成処理の戻り値が0以外である場合は、処理中で何らかのエラーが発生している場合となります。
    // (出力ファイルは生成されません)
    // 戻り値、及びlastMessageメソッドにより取得できるエラーメッセージから原因を特定し対応します。
    //
    // 【V8.0.0】VirtualFileクラス の廃止
    //         intra-mart API の VirtualFileクラスが廃止されました。
    //         代替クラスとして、PublicStorageクラス を使用します。
```

```
171     //          使用方法は、VirtualFileクラスと同じです。
172     //
173     if(resultCode == 0){
174         // 結果PDFファイルのダウンロード
175         var pdfpath = new PublicStorage(outPdfPath);
176         Module.download.send(pdfpath, outPdfName);
177     }
178     else{
179         Module.alert.reload("SYSTEM.ERR", "(" + resultCode + ")" + pdf.getMessage());
180     }
181 }
```

記述が完了したら %HOME_PATH%/jssp/src/pdfd/tutorial ディレクトリに、
docsample.js というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

3. 認可・ルーティング設定

intra-mart Accel Platform の認可及びルーティング設定に従い、以下の設定をして下さい。

- path属性：任意のURL文字列
- page属性：pdfd/tutorial/docsample

4. 画面表示・プログラム実行

設定したURLにアクセスすると、以下の画面が表示されます。

「PDF作成」ボタンをクリックすると、PDFファイルが作成され処理終了後にダウンロードが開始されます。
実行エラーが発生した場合には、エラーメッセージの内容に従いJavaScriptファイルもしくはhtmlファイルを修正してください。

5. 確認

プログラムが正しく実行されると Public Storage の pdfd/tutorial/ に PDFファイルが作成されています。
このファイルがPDFのビューア (AdobeReaderなど) で正しく表示できれば、すべての処理が正しく行われたことになります

連票形式

項目

連票形式のPDFファイルを作成するためのスクリプトプログラムを作成します。スクリプト開発では、htmlファイルとJavaScriptファイルを作成する必要があります。

コラム

文字コードは UTF-8 でファイルを保存して下さい。

1. 入力画面の作成

テキストエディタを起動して、以下のHTMLを記述します。

```
<!--
// CSVCELAサンプル(PDF-Desiner V8.0.0)
// IOCela帳票データをコード内部で生成しPDF帳票ファイルを生成します。
// PDFファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
   -->
<DIV align="center" style="center; padding-top: 25px;">
  <P><FONT size="+2">チュートリアルサンプル(IOCela)</FONT></P>
  <TABLE border="1">
    <TR>
      <TH align="center" style="padding: 5px 10px;" nowrap>
        出力PDFファイルは、<BR>Public Storageの[pdfd/tutorial]<BR>
        フォルダ下に作成され、処理終了後に自動ダウンロードされます。
      </TH>
    </TR>
    <TR>
      <TH align="center" style="padding: 5px 10px;" nowrap>
        処理実行は下のボタンをクリックします。
      </TH>
    </TR>
    <TR>
      <TD align="center" style="padding: 5px 10px;" nowrap>
        <IMART type="form" action="makePDF">
          <INPUT type="submit" value=" PDF作成 ">
        </IMART>
      </TD>
    </TR>
  </TABLE>
</DIV>
```

記述が完了したら %HOME_PATH%/jssp/src/pdfd/tutorial ディレクトリを作成し、celasample.html というファイル名で保存して下さい。

この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

2. 入力画面処理の作成

次にJavaScriptファイルを作成します。

“//”から始まる行は、コメントですので無視して記述頂いても問題ありません。

```
//---
// CSVCELAサンプル(PDF-Desiner V8.0.0)
//---
// IOCela帳票データをコード内部で生成しPDF帳票ファイルを生成します。
// PDFファイルへは、文書情報／セキュリティ情報を付加し、出力しています。
function makePDF(request){

    //-----
    // 出力ファイルパスの設定
    //-----
    // 出力ファイルは、Storage上のPublicディレクトリ以下の任意の位置に出力できます。
    //
    //    Publicディレクトリとは、
    //        %PUBLIC_STORAGE_PATH%/public/storage/ までを指しています。
    //
    // また、ファイル名は PublicStorageクラスを使用しStorage上に同一ファイルがないか確認し
    // 同一ファイルが存在する場合は、ファイル名に"_"(アンダーバー)+数値"を付加しています。
    //
    // *** このサンプルでは完全な一意性は確保できません。***
    //
    var sessionid  = Client.identifier();   // セッションIDの取得
    var dirPath    = "pdfd/tutorial/";      // 出力フォルダ
    var prefix     = "designer";            // 出力ファイル接頭文字
    var suffix     = ".pdf";                // 出力ファイル拡張子
    var outPdfName = prefix + "_" + sessionid + suffix;
    var outPdfPath = dirPath + outPdfName;  // pdfd/tutorial/nouhinken_[sessionid].pdf

    var ps = new PublicStorage(outPdfPath);
    var i = 1;
    while (ps.exists()) {
        outPdfName = prefix + "_" + sessionid + "_" + i + suffix;
        outPdfPath = dirPath + outPdfName;
        ps = new PublicStorage(outPdfPath);
        i++;
    }

    //-----
    // インスタンス生成 (V8.0.0から変更あり)
    //-----
    // DEFファイルを指定
    //
    // 【V8.0.0】ファイルパス指定方法の変更
    //          Storage の Publicディレクトリ からの相対パスを指定します。
    //
    var pdf = new IOCela("pdfd/tutorial/designer.def");

    //-----
    // 文書情報設定 (V7.x から変更なし)
    //-----
    // 文書情報セット（各項目最大255文字まで）
    //
    // defineTitle(String)       タイトルの設定
    // defineSubTitle(String)    サブタイトルの設定
    // defineAuthor(String)      作成者の設定
    // defineApplication(String) 作成アプリケーション名の設定
    //
```

```
57      //
58      pdf.defineTitle("PDFデザイナー体験");
59      pdf.defineAuthor("IM 太郎");
60
61      //-----
62      // セキュリティ設定 (V7.x から変更なし)
63      //-----
64      // セキュリティ情報セット（パスワードは最大32文字まで）
65      //
66      // <パスワード設定>
67      //    setOpenPassword(String)     オープンパスワード(32文字まで)
68      //    setSecurityPassword(String) セキュリティパスワード(32文字まで)
69      //
70      // <印刷許可設定>
71      //    printSecurity("PRINT_ENABLE")  印刷許可
72      //    printSecurity("PRINT_DISABLE") 印刷不許可
73      //
74      // <変更許可設定>
75      //    modifySecurity("MODIFY_DISABLE")            変更不許可
76      //    modifySecurity("MODIFY_ALL")                変更許可
77      //                                        (ページの抽出を除くすべての変更を許可)
78      //    modifySecurity("MODIFY_FORM_AND_ANNOTATION") 変更許可
79      //                                        ("注釈の作成","フォームフィールドの入力",
80      //                                         "既存の署名フィールドに署名"を許可)
81      //    modifySecurity("MODIFY_FORM_AND_ASSEMBLY")   変更許可
82      //                                        ("ページレイアウト",
83      //                                         "フォームフィールドの入力",
84      //                                         "既存の署名フィールドに署名"を許可)
85      //
86      // <テキスト文字抽出許可及びアクセシビリティ許可設定>
87      //    pdf.copySecurity("COPY_AND_ACCESSBILITY_DISABLE") 不許可
88      //    pdf.copySecurity("COPY_AND_ACCESSBILITY_ENABLE")  許可
89      //
90      pdf.setSecurityPassword("secpasswd");
91      pdf.printSecurity("PRINT_DISABLE");
92      pdf.modifySecurity("MODIFY_DISABLE");
93      pdf.copySecurity("COPY_AND_ACCESSBILITY_DISABLE");
94
95      //-----
96      // ページデータの生成 (V7.x から変更なし)
97      //-----
98      // 本チュートリアルでは、レコードオブジェクト形式でのデータ設定を実施します。
99      // またIODocレイアウトの重ね合わせを実施し、
100     // 内部データをデータオブジェクト形式で設定します。
101     //
102     // CSVファイルセット
103     // ファイル内容はDEFファイル上のデータ形式設定に沿って各カラムに値が挿入されます。
104     //
105     pdf.setCSV("pdfd/tutorial/designer_data.csv");
106
107     // CSVデータをコード内部でセットする場合では、
108     // 以下のようにsetRecordに1レコード分のデータ文字列をセットします。
109   /*
110     pdf.setRecord("株式会社NTTデータイントラマート    ",
111                   "川崎太郎    ",
112                   "2008/10/1    ",
113                   "株式会社川崎商事    ",
```

```
114             "PDFデザイナー　",
115             "490000　",
116             "1　",
117             "備考1　",
118             "備考2");
119 */
120
121   //-----
122   // PDF出力処理 (V8.0.0 から変更あり)
123   //-----
124   // PDFファイルへの出力処理が実行されます。
125   // 正常に処理が完了した場合には指定されたPDFファイル名に該当の文書が作成されます。
126   //
127   // 【V8.0.0】ファイルパス指定方法の変更
128   //         Storage の Publicディレクトリ からの相対パスを指定します。
129   //
130   resultCode = pdf.toPDF(outPdfPath);
131
132   //-----
133   // 終了処理 (V8.0.0 から変更あり)
134   //-----
135   // PDF作成処理の戻り値が0以外である場合は、処理中で何らかのエラーが発生している場合となります。
136   // (出力ファイルは生成されません)
137   // 戻り値、及びlastMessageメソッドにより取得できるエラーメッセージを参考に、原因を特定し対応します。
138   //
139   // 【V8.0.0】VirtualFileクラス の廃止
140   //         intra-mart API の VirtualFileクラスが廃止されました。
141   //         代替クラスとして、PublicStorageクラス を使用します。
142   //         使用方法は、VirtualFileクラスと同じです。
143   //
144   if(resultCode == 0){
145     // 結果PDFファイルのダウンロード
146     var pdfpath = new PublicStorage(outPdfPath);
147     Module.download.send(pdfpath, outPdfName);
148   }
149   else{
150     Module.alert.reload("SYSTEM.ERR", "(" + resultCode + ")" + pdf.getMessage());
151   }
152 }
```

記述が完了したら %HOME_PATH%/jssp/src/pdfd/tutorial ディレクトリに、
celasample.js というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

3. 認可・ルーティング設定

intra-mart Accel Platform の認可及びルーティング設定に従い、以下の設定をして下さい。

- path属性：任意のURL文字列
- page属性：pdfd/tutorial/celasample

4. 画面表示・プログラム実行

設定したURLにアクセスすると、以下の画面が表示されます。

「PDF作成」ボタンをクリックすると、PDFファイルが作成され処理終了後にダウンロードが開始されます。
実行エラーが発生した場合には、エラーメッセージの内容に従いJavaScriptファイルもしくはhtmlファイルを修正してください。

5. 確認

プログラムが正しく実行されると Public Storage の pdfd/tutorial/ に PDFファイルが作成されています。
このファイルがPDFのビューア (AdobeReaderなど) で正しく表示できれば、すべての処理が正しく行われたことになります

JavaEE 開発モデル

単票形式、連表形式について、実際に帳票を出力するまでの過程を説明します。

単票形式

項目

単票形式のPDFファイルを作成するための JSPプログラムを作成します。
JSPプログラムに記述されたPDF生成処理は、アプリケーション上で実行されます。
ここでは、JSPプログラムにPDF生成処理を記載していますが、JSPプログラムから分離してJavaプログラムとして作成することも可能です。

コラム

文字コードは UTF-8 でファイルを保存して下さい。

1. 入力画面処理の作成

入力画面処理のプログラムを記述します。

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<DIV align="center" style="center; padding-top: 25px;">
   <P><FONT size="+2">チュートリアルサンプル(IODoc)</FONT></P>
   <FORM action="pdfd/javaee/tutorial/docsample_act" method="POST">
      <TABLE border="1">
         <TR>
            <TH align="center" style="padding: 5px 10px;" nowrap>
               出力PDFファイルディレクトリ: Public Storage の [pdfd/tutorial]
            </TH>
         </TR>
         <TR>
            <TH align="center" style="padding: 5px 10px;" nowrap>
               下のボタンをクリックすることでPDF生成を開始します。
            </TH>
         </TR>
         <TR>
            <TD align="center" style="padding: 5px 10px;" nowrap>
               <input type="submit" value=" PDF生成 " />
            </TD>
         </TR>
      </TABLE>
   </FORM>
</DIV>
```

記述が完了したら %HOME_PATH%/view/pdfd/tutorial ディレクトリを作成し、
docsample.jsp というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

2. 出力画面処理の作成

次に出力画面処理のプログラムを記述します。

“//”から始まる行は、コメントですので無視して記述頂いても問題ありません。

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ page import="jp.co.intra_mart.foundation.service.client.file.PublicStorage" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.PDFLibSecurity" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.net.CSVDoc" %>
<%
//---
// CSVDOCサンプル(PDF-Desiner V8.0.0)
//---
// 単票用レイアウトファイルと CSV 形式のデータファイル
// からIOD中間ファイル、PDFファイルを作成するための機能を提供します。
//
// 本チュートリアルでは、単票用レイアウトファイルに
// DATデータの設定後、PDFファイルを生成しています。
//
    //-----
    // 出力ファイルパスの設定
    //-----
    // 出力ファイルは、Storage上のPublicディレクトリ以下の任意の位置に出力できます。
    //
    //     Publicディレクトリとは、
    //         %PUBLIC_STORAGE_PATH%/public/storage/ までを指しています。
    //
    // また、ファイル名は PublicStorageクラスを使用しStorage上に同一ファイルがないか確認し
    // 同一ファイルが存在する場合は、ファイル名に"_"(アンダーバー)+数値"を付加しています。
    //
    // *** このサンプルでは完全な一意性は確保できません。***
    //
    String outpath    = "pdfd/tutorial/"; // 出力フォルダ
    String prefix     = "nouhinkensa";    // 出力ファイル接頭文字
    String suffix     = ".pdf";           // 出力ファイル拡張子
    String outPdfPath = outpath + prefix + suffix;

    PublicStorage ps = new PublicStorage(outPdfPath);
    int i = 1;
    while (ps.exists()) {
        outPdfPath = outpath + prefix + "_" + i + suffix;
        ps = new PublicStorage(outPdfPath);
        i++;
    }

    //-----
    // インスタンス生成 (V8.0.0から変更あり)
    //-----
    // メモリオブジェクト方式のため、入力IODのみ指定。
    // CSVファイル形式の場合はCCDファイルも指定します。
    //
    // 【V8.0.0】ファイルパス指定方法の変更
    //           Storage の Publicディレクトリ からの相対パスを指定します。
    //
    CSVDoc pdf = new CSVDoc("pdfd/tutorial/nouhinkensa.iod", "");

    //-----
    // 文書情報設定 (V7.x から変更なし)
```

```
//-----
// 文書情報セット（各項目最大255文字まで）
//
// defineTitle(String)       タイトルの設定
// defineSubTitle(String)    サブタイトルの設定
// defineAuthor(String)      作成者の設定
// defineApplication(String) 作成アプリケーション名の設定
//
//
pdf.defineTitle("納品書兼検査票");
pdf.defineAuthor("IM 太郎");

//-----
// セキュリティ設定 (V7.x から変更なし)
//-----
// セキュリティ情報セット（パスワードは最大32文字まで）
//
// <パスワード設定>
//    setOpenPassword(String)     オープンパスワード(32文字まで)
//    setSecurityPassword(String) セキュリティパスワード(32文字まで)
//
// <印刷許可設定>
//    printSecurity(PDFLibSecurity.PRINT_ENABLE)  印刷許可
//    printSecurity(PDFLibSecurity.PRINT_DISABLE) 印刷不許可
//
// <変更許可設定>
//    modifySecurity(PDFLibSecurity.MODIFY_DISABLE)
//       変更不許可
//    modifySecurity(PDFLibSecurity.MODIFY_ALL)
//       変更許可 (ページの抽出を除くすべての変更を許可)
//    modifySecurity(PDFLibSecurity.MODIFY_FORM_AND_ANNOTATION)
//       変更許可 ("注釈の作成","フォームフィールドの入力",
//                 "既存の署名フィールドに署名"を許可)
//    modifySecurity(PDFLibSecurity.MODIFY_FORM_AND_ASSEMBLY)
//       変更許可 ("ページレイアウト", "フォームフィールドの入力",
//                 "既存の署名フィールドに署名"を許可)
//
// <テキスト文字抽出許可及びアクセシビリティ許可設定>
//    copySecurity(PDFLibSecurity.COPY_AND_ACCESSBILITY_DISABLE) 不許可
//    copySecurity(PDFLibSecurity.COPY_AND_ACCESSBILITY_ENABLE)  許可
//
pdf.setSecurityPassword("secpasswd");
pdf.printSecurity(PDFLibSecurity.PRINT_DISABLE);
pdf.modifySecurity(PDFLibSecurity.MODIFY_DISABLE);
pdf.copySecurity(PDFLibSecurity.COPY_AND_ACCESSBILITY_DISABLE);

//-----
// ページデータの生成 (V7.x から変更なし)
//-----
// 本チュートリアルでは、メモリオブジェクト形式でのデータ設定を実施します。
// CSV/DATファイルオブジェクトでデータを与える場合には、
// ここでそれぞれ入力ファイルを設定します。
//
// 埋め込み識別子及びデータ
// (DBデータ検索等により取得、又はコード内で埋め込みデータを生成することの可能)
//
String[][] doc_data = {
```

```
            {"kyakusaki","OrderComNo","nouhin_No","tantou","nouhinsaki",
             "tyuumon_No","hinmei_code","hinmei","h_memo","syukka_day",
             "suuryou","tani","tanka","j_memo","nouki",
             "shiji_suuryou","nounyu_suuryou","konpou_suuryou","zei","zeinuki",
             "zeikomi","BarCode1","bar1","bar2","bar3"},
            {"NTTデータイントラマート","001","001-001","IM 太郎","IM商事",
            "C-001-001","YPDFAUTO-001","IM-PDFオートコンバータ","","2004/07/01",
             "2","式","1000000","","2008/07/05",
             "2","2","2","100000","2000000",
             "2100000","CODE39","CODE39","CODE39","CODE39"}
    };

    //-----
    // テキスト関連識別子データ埋め込み (V7.x から変更なし)
    //-----
    // 通常テキスト文字列を埋め込みします。
    // (※複数行カラムデータの識別子名へは、[識別子#行番号]と編集してセットします)
    //
    for(i = 0;i < doc_data[0].length;i++) {
        pdf.setData(doc_data[0][i],doc_data[1][i]);
    }

    //-----
    // PDF出力処理 (V8.0.0 から変更あり)
    //-----
    // PDFファイルへの出力処理が実行されます。
    // 正常に処理が完了した場合には指定されたPDFファイル名に該当の文書が作成されます。
    //
    // 【V8.0.0】ファイルパス指定方法の変更
    //          Storage の Publicディレクトリ からの相対パスを指定します。
    //
    int resultCode = pdf.makePDF(outPdfPath);

    //-----
    // 終了処理 (V7.x から変更なし)
    //-----
    // PDF作成処理の戻り値が0以外である場合は、処理中で何らかのエラーが発生している場合となります。
    // (出力ファイルは生成されません)
    // 戻り値、及びlastMessageメソッドにより取得できるエラーメッセージから原因を特定し対応します。
    //
    String resultMessage = "";
    if(resultCode == 0){
        resultMessage = "Success !!";
    }
    else{
        resultMessage = pdf.lastMessage();
    }

// 以下Webブラウザ出力HTMLレコードです。
// 当JSPを呼び出し時に上記IOCela帳票からのPDFファイル生成が実施され、
// 正常に完了した場合には、出力PDFファイルをダウンロードする為のリンク
// が表示されます。
// 出力には処理戻り値、メッセージ取得内容を含みます。
 %>
<DIV align="center" style="center; padding-top: 25px;">
    <P><FONT size="+2">チュートリアルサンプル(IODoc)</FONT></P>
    <FORM action="pdfd/javaee/tutorial/outfile" method="POST">
```

```
      <TABLE border="1">
        <TR>
          <TH align="right" style="padding: 5px 10px;" nowrap>
            出力PDFファイル
          </TH>
          <TD align="left" style="padding: 5px 10px;" nowrap>
            <%= outPdfPath %>
          </TD>
        </TR>
        <TR>
          <TH align="right" style="padding: 5px 10px;" nowrap>
            戻り値
          </TH>
          <TD align="left" style="padding: 5px 10px;" nowrap>
            <%= resultCode %>
          </TD>
        </TR>
        <TR>
          <TH align="right" style="padding: 5px 10px;" nowrap>
            メッセージ
          </TH>
          <TD align="left" style="padding: 5px 10px;" nowrap>
            <%= resultMessage %>
          </TD>
        </TR>
<% if(resultCode == 0) { %>
        <TR>
          <TD colspan="2" align="center" style="padding: 5px 10px;" nowrap>
            <INPUT type="hidden" name="file" value="<%= outPdfPath %>" />
            <INPUT type="submit" value=" download " />
          </TD>
        </TR>
<% } %>
      </TABLE>
    </FORM>
</DIV>
```

記述が完了したら %HOME_PATH%/view/pdfd/tutorial ディレクトリに、
docsample_act.jsp というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

3. 認可・ルーティング設定

intra-mart Accel Platform の認可及びルーティング設定に従い、以下の設定をして下さい。

<入力画面処理>

- path属性：任意のURL文字列
- page属性：WEB-INF/view/pdfd/tutorial/docsample.jsp

<出力画面処理>

- path属性：任意のURL文字列
- page属性：WEB-INF/view/pdfd/tutorial/docsample_act.jsp

4. 画面表示・プログラム実行

設定したURLにアクセスすると、以下の画面が表示されます。

「PDF作成」ボタンをクリックすると、PDFファイルが作成され処理終了後にダウンロードが開始されます。
実行エラーが発生した場合には、エラーメッセージの内容に従いJSPファイルを修正してください。

5. 確認

プログラムが正しく実行されると Public Storage の pdfd/tutorial/ に PDFファイルが作成されています。
このファイルがPDFのビューア (AdobeReaderなど) で正しく表示できれば、すべての処理が正しく行われたことになります

連票形式

項目

連票形式のPDFファイルを作成するための JSPプログラムを作成します。
JSPプログラムに記述されたPDF生成処理は、アプリケーション上で実行されます。
ここでは、JSPプログラムにPDF生成処理を記載していますが、JSPプログラムから分離してJavaプログラムとして作成することも可能です。

コラム

文字コードは UTF-8 でファイルを保存して下さい。

1. 入力画面処理の作成

入力画面処理のプログラムを記述します。

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<DIV align="center" style="center; padding-top: 25px;">
   <P><FONT size="+2">チュートリアルサンプル(IOCela)</FONT></P>
   <FORM action="pdfd/javaee/tutorial/celasample_act" method="POST">
      <TABLE border="1">
         <TR>
            <TH align="center" style="padding: 5px 10px;" nowrap>
               出力PDFファイルディレクトリ: Public Storage の [pdfd/tutorial]
            </TH>
         </TR>
         <TR>
            <TH align="center" style="padding: 5px 10px;" nowrap>
               下のボタンをクリックすることでPDF生成を開始します。
            </TH>
         </TR>
         <TR>
            <TD align="center" style="padding: 5px 10px;" nowrap>
               <input type="submit" value=" PDF生成 " />
            </TD>
         </TR>
      </TABLE>
   </FORM>
</DIV>
```

記述が完了したら %HOME_PATH%/view/pdfd/tutorial ディレクトリを作成し、
celasample.jsp というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

2. 出力画面処理の作成

次に出力画面処理のプログラムを記述します。

“//”から始まる行は、コメントですので無視して記述頂いても問題ありません。

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ page import="jp.co.intra_mart.foundation.service.client.file.PublicStorage" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.PDFLibSecurity" %>
<%@ page import="jp.co.intra_mart.product.pdfmaker.net.CSVCela" %>
<%
//---
// CSVCELAサンプル(PDF-Desiner V8.0.0)
//---
// IOCELA帳票データをコード内部で生成しPDF帳票ファイルを生成します。
// PDFファイルは、文書情報／セキュリティ情報を付加し、出力しています。
//

    //-----
    // 出力ファイルパスの設定
    //-----
    // 出力ファイルは、Storage上のPublicディレクトリ以下の任意の位置に出力できます。
    //
    //     Publicディレクトリとは、
    //         %PUBLIC_STORAGE_PATH%/public/storage/ までを指しています。
    //
    // また、ファイル名は PublicStorageクラスを使用しStorage上に同一ファイルがないか確認し
    // 同一ファイルが存在する場合は、ファイル名に"_"(アンダーバー)+数値"を付加しています。
    //
    // *** このサンプルでは完全な一意性は確保できません。***
    //
    String outpath    = "pdfd/tutorial/"; // 出力フォルダ
    String prefix     = "designer";    // 出力ファイル接頭文字
    String suffix     = ".pdf";           // 出力ファイル拡張子
    String outPdfPath = outpath + prefix + suffix;

    PublicStorage ps = new PublicStorage(outPdfPath);
    int i = 1;
    while (ps.exists()) {
        outPdfPath = outpath + prefix + "_" + i + suffix;
        ps = new PublicStorage(outPdfPath);
        i++;
    }

    //-----
    // インスタンス生成 (V8.0.0から変更あり)
    //-----
    // 引数として連票用レイアウトファイルパスを指定(必須)
    //
    // 【V8.0.0】ファイルパス指定方法の変更
    //           Storage の Publicディレクトリ からの相対パスを指定します。
    //
    CSVCela pdf = new CSVCela("pdfd/tutorial/designer.def");

    //-----
    // 文書情報設定 (V7.x から変更なし)
    //-----
    // 文書情報セット（各項目最大255文字まで）
    //
```

```
// defineTitle(String)       タイトルの設定
// defineSubTitle(String)    サブタイトルの設定
// defineAuthor(String)      作成者の設定
// defineApplication(String) 作成アプリケーション名の設定
//
//
pdf.defineTitle("PDFデザイナー体験");
pdf.defineAuthor("IM 太郎");

//-----
// セキュリティ設定 (V7.x から変更なし)
//-----
// セキュリティ情報セット（パスワードは最大32文字まで）
//
// <パスワード設定>
//    setOpenPassword(String)     オープンパスワード(32文字まで)
//    setSecurityPassword(String) セキュリティパスワード(32文字まで)
//
// <印刷許可設定>
//    printSecurity(PDFLibSecurity.PRINT_ENABLE)  印刷許可
//    printSecurity(PDFLibSecurity.PRINT_DISABLE) 印刷不許可
//
// <変更許可設定>
//    modifySecurity(PDFLibSecurity.MODIFY_DISABLE)
//       変更不許可
//    modifySecurity(PDFLibSecurity.MODIFY_ALL)
//       変更許可 (ページの抽出を除くすべての変更を許可)
//    modifySecurity(PDFLibSecurity.MODIFY_FORM_AND_ANNOTATION)
//       変更許可 ("注釈の作成","フォームフィールドの入力",
//                "既存の署名フィールドに署名"を許可)
//    modifySecurity(PDFLibSecurity.MODIFY_FORM_AND_ASSEMBLY)
//       変更許可 ("ページレイアウト", "フォームフィールドの入力",
//                "既存の署名フィールドに署名"を許可)
//
// <テキスト文字抽出許可及びアクセシビリティ許可設定>
//    copySecurity(PDFLibSecurity.COPY_AND_ACCESSBILITY_DISABLE) 不許可
//    copySecurity(PDFLibSecurity.COPY_AND_ACCESSBILITY_ENABLE)  許可
//
pdf.setSecurityPassword("secpasswd");
pdf.printSecurity(PDFLibSecurity.PRINT_DISABLE);
pdf.modifySecurity(PDFLibSecurity.MODIFY_DISABLE);
pdf.copySecurity(PDFLibSecurity.COPY_AND_ACCESSBILITY_DISABLE);

//-----
// CSVデータファイル設定 (V8.0.0 から変更あり)
//-----
// ファイル内容はDEFファイル上のデータ形式設定に沿って各カラムに値が挿入されます。
//
// 【V8.0.0】ファイルパス指定方法の変更
//          Storage の Publicディレクトリ からの相対パスを指定します。
//
pdf.setCSV("pdfd/tutorial/designer_data.csv");

//-----
// レコードデータ設定 (V7.x から変更なし)
//-----
// CSVデータファイルの代わりにCSV形式のレコードデータを指定する。
```

```
    // 1回で1行分のデータを設定することが可能です。
    // (複数行の設定をする場合は、複数回メソッドを呼び出して下さい。)
    //
//  for(int row = 1; row <= 120; row++) {
//    pdf.setRecord(String.format("%d_data;%d;%d;%d;%d;%d",
//                                row,
//                                row+1,
//                                row+2,
//                                row+3,
//                                row+4,
//                                row+5));
//  }

    //-----
    // PDF出力処理 (V8.0.0 から変更あり)
    //-----
    // PDFファイルへの出力処理が実行されます。
    // 正常に処理が完了した場合には指定されたPDFファイル名に該当の文書が作成されます。
    //
    //【V8.0.0】ファイルパス指定方法の変更
    //          Storage の Publicディレクトリ からの相対パスを指定します。
    //
    int resultCode = pdf.makePDF(outPdfPath);

    //-----
    // 終了処理 (V7.x から変更なし)
    //-----
    // PDF作成処理の戻り値が0以外である場合は、処理中で何らかのエラーが発生している場合となります。
    // (出力ファイルは生成されません)
    // 戻り値、及びlastMessageメソッドにより取得できるエラーメッセージから原因を特定し対応します。
    //
    String resultMessage = "";
    if(resultCode == 0){
        resultMessage = "Success !!";
    }
    else{
        resultMessage = pdf.lastMessage();
    }

// 以下Webブラウザ出力HTMLレコードです。
// 当JSPを呼び出し時に上記IOCela帳票からのPDFファイル生成が実施され、
// 正常に完了した場合には、出力PDFファイルをダウンロードする為のリンクが表示されます。
// 出力には処理戻り値、メッセージ取得内容を含みます。
 %>
<DIV align="center" style="center; padding-top: 25px;">
    <P><FONT size="+2">チュートリアルサンプル(IOCela)</FONT></P>
    <FORM action="pdfd/javaee/tutorial/outfile" method="POST">
        <TABLE border="1">
            <TR>
                <TH align="right" style="padding: 5px 10px;" nowrap>
                    出力PDFファイル
                </TH>
                <TD align="left" style="padding: 5px 10px;" nowrap>
                    <%= outPdfPath %>
                </TD>
            </TR>
            <TR>
```

```
            <TH align="right" style="padding: 5px 10px;" nowrap>
              戻り値
            </TH>
            <TD align="left" style="padding: 5px 10px;" nowrap>
              <%= resultCode %>
            </TD>
          </TR>
          <TR>
            <TH align="right" style="padding: 5px 10px;" nowrap>
              メッセージ
            </TH>
            <TD align="left" style="padding: 5px 10px;" nowrap>
              <%= resultMessage %>
            </TD>
          </TR>
<% if(resultCode == 0) { %>
          <TR>
            <TD colspan="2" align="center" style="padding: 5px 10px;" nowrap>
              <INPUT type="hidden" name="file" value="<%= outPdfPath %>" />
              <INPUT type="submit" value=" download " />
            </TD>
          </TR>
<% } %>
        </TABLE>
    </FORM>
</DIV>
```

記述が完了したら %HOME_PATH%/view/pdfd/tutorial ディレクトリに、
celasample_act.jsp というファイル名で保存して下さい。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意して下さい。

3. 認可・ルーティング設定

intra-mart Accel Platform の認可及びルーティング設定に従い、以下の設定をして下さい。

<入力画面処理>

- path属性：任意のURL文字列
- page属性：WEB-INF/view/pdfd/tutorial/celasample.jsp

<出力画面処理>

- path属性：任意のURL文字列
- page属性：WEB-INF/view/pdfd/tutorial/celasample_act.jsp

4. 画面表示・プログラム実行

設定したURLにアクセスすると、以下の画面が表示されます。

「PDF作成」ボタンをクリックすると、PDFファイルが作成され処理終了後にダウンロードが開始されます。
実行エラーが発生した場合には、エラーメッセージの内容に従いJavaScriptファイルもしくはhtmlファイルを修正してください。

5. 確認

プログラムが正しく実行されると Public Storage の pdfd/tutorial/ に PDFファイルが作成されています。
このファイルがPDFのビューア (AdobeReaderなど) で正しく表示できれば、すべての処理が正しく行われたことになります

サンプルプログラム

本製品には、PDFデザイナーのAPIの使用方法を説明したサンプルプログラムが同梱されています。
サンプルプログラムはPDFデザイナーインストール時に一緒にインストールされます。

ここでは、サンプルプログラムの実行方法と内容について説明します。

サンプルプログラム･データの保存位置

項目

サンプルプログラムの保存位置

スクリプト開発モデル

以下のフォルダに保存されています。

- %HOME_PATH%/jssp/src/pdfd/

Dir	html/js	内容
sample	celacsv	CSVファイルを用いて連票形式のPDFを作成するサンプル
	celacsvdat	CSVファイルを用いてレイアウトを重ね合わせるサンプル
	celarec	レコードデータを用いて連票形式のPDFを作成するサンプル
	celarecobj	メモリデータを用いてレイアウトを重ね合わせるサンプル
	doccsv	CSVファイルを用いて単票形式のPDFを作成するサンプル
	docdat	DATファイルを用いて単票形式のPDFを作成するサンプル
	docobj	メモリデータを用いて単票形式のPDFを作成するサンプル
	integration	結合PDFファイルを作成するサンプル
tutorial	celasample	CSVファイルを用いて連票形式のPDFを作成するサンプル
	docsample	CSVファイルを用いてレイアウトを重ね合わせるサンプル
	iointegration	結合PDFファイルを作成するサンプル

JavaEE開発モデル

以下のフォルダに保存されています。

- %HOME_PATH%/view/pdfd/

Dir	jsp	内容
sample	celacsv	CSVファイルを用いて連票形式のPDFを作成するサンプル
	celacsvdat	CSVファイルを用いてレイアウトを重ね合わせるサンプル
	celarec	レコードデータを用いて連票形式のPDFを作成するサンプル

Dir	jsp	内容
	celarecobj	メモリデータを用いてレイアウトを重ね合わせるサンプル
	doccsv	CSVファイルを用いて単票形式のPDFを作成するサンプル
	docdat	DATファイルを用いて単票形式のPDFを作成するサンプル
	docobj	メモリデータを用いて単票形式のPDFを作成するサンプル
	integration	結合PDFファイルを作成するサンプル
tutorial	celasample	CSVファイルを用いて連票形式のPDFを作成するサンプル
	docsample	CSVファイルを用いてレイアウトを重ね合わせるサンプル
	iointegration	結合PDFファイルを作成するサンプル

サンプルデータ(レイアウトなど)の保存位置

以下のフォルダに保存されています。

- %PUBLIC_STORAGE_PATH%/public/storage/pdfd/

Dir	内容
tutorial	チュートリアル用のデータ
webdoc	単票形式のPDFファイル作成サンプル用のデータ
webcela	連票形式のPDFファイル作成サンプル用のデータ
integration	結合サンプル用のデータ

サンプルプログラムの説明

PDFデザイナーはさまざまな形式のデータを指定してPDFファイルを作成することができます。

単票形式については、以下3種類の方法でデータを指定しPDFファイルを作成することができます。

1. CSVファイルを指定してPDFファイルを作成する。
2. DATファイルを指定してPDFファイルを作成する。
3. メモリデータを指定してPDFファイルを作成する。

連票形式については、以下4種類の方法でデータを指定しPDFファイルを作成することができます。

1. CSVファイルを指定してPDFファイルを作成する。
2. レコードデータを指定してPDFファイルを作成する。
3. CSVファイルと単票形式のレイアウトファイルを重ね合わせてPDFファイルを作成する。
4. メモリデータと単票形式のレイアウトファイルを重ね合わせてPDFファイルを作成する。

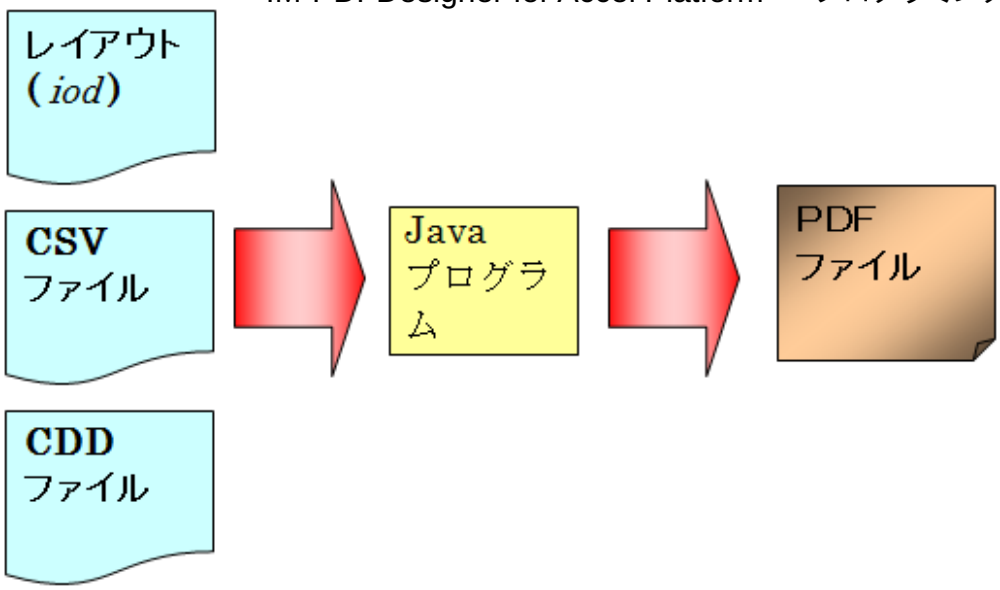
結合については、単票形式、連票形式で作成した中間ファイル (IODファイル) を、1枚の PDFファイル として結合することができます。

以下、サンプルプログラムについて説明します。

CSVファイルを用いて単票形式のPDFを作成するサンプル

帳票レイアウトファイルとCSVファイルを指定して、PDFファイルを作成します。
CSVファイルと連携する場合、帳票レイアウトとCSVファイルのデータを関連付けるキーマップ（cddファイル）が必要となります。
CDDファイルについては、CDDエディタを利用すると簡単に作成することができます。
CDDエディタの使い方に関しては、専用マニュアル tool/document/cddedit.pdfをご覧ください。

帳票レイアウト、CSVファイル、CDDファイルからPDFファイルを作成します。



CSVファイルを用いて単票形式のPDFを作成する方法については、以下のサンプルプログラムをご覧ください。

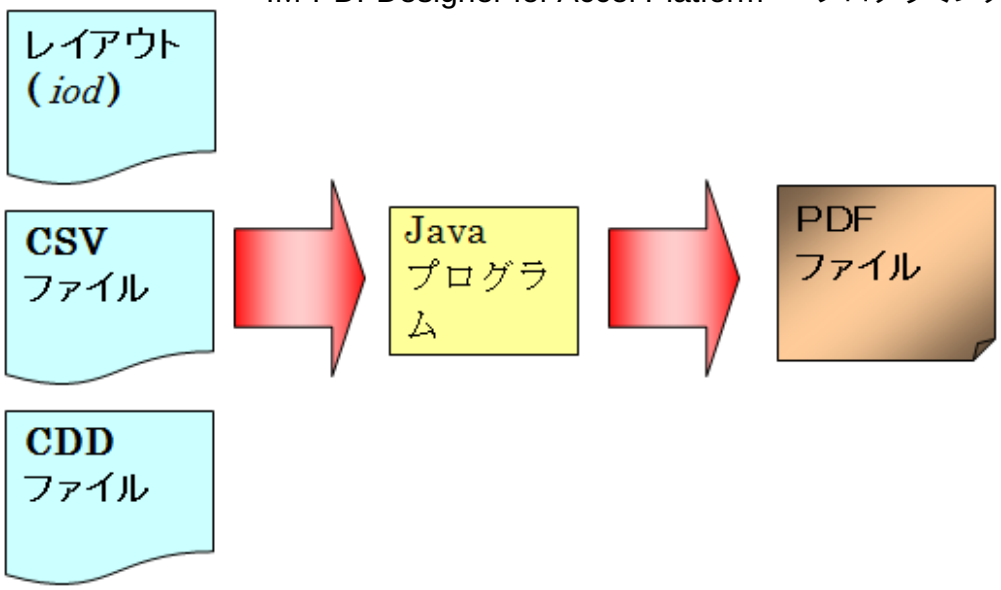
JavaEE開発モデル	jsp/pdfd/sample/doccsv_act.jsp
スクリプト開発モデル	jssp/src/pdfd/sample/doccsv.js

DATファイルを用いて単票形式のPDFを作成するサンプル

帳票レイアウトファイルとDATファイルを指定して、PDFファイルを作成します。
DATファイルとは、帳票レイアウトで指定した属性名と、その属性にセットする値を記述したテキスト形式のファイルです。
以下にDATファイルのサンプルを示します。

※DATファイルサンプル
同名の属性名が複数存在する場合、#(連番)の形式で指定可能です。

```
Kyakusaki 株式会社 yss
NohinshoNo 100
Hinmei#1 EBW-Z1011
Hinmei#2 EBW-Z1210
Hinmei#3 EBW-Z1411
Hinmei#4 EBW-Z1612
Hinmei#5 EBW-Z1712
Hinmei#6 EBW-Z2014
Suryo#1 5
Suryo#2 5
```

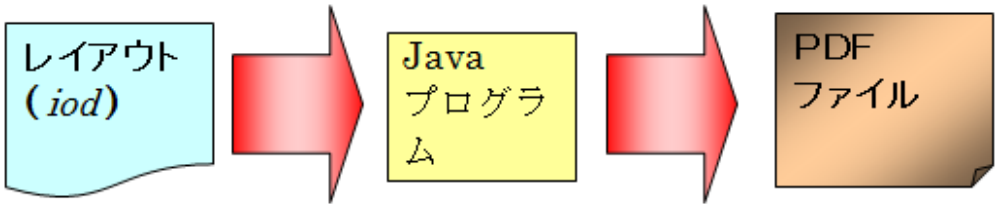


DATファイルを用いて単票形式のPDFを作成する方法については、以下のサンプルプログラムをご覧ください。

JavaEE開発モデル	jsp/pdfd/sample/docdat_act.jsp
スクリプト開発モデル	jssp/src/pdfd/sample/docdat.js

メモリデータを用いて単票形式のPDFを作成するサンプル

帳票レイアウトファイルを作成し、プログラム内部でデータを設定してPDFファイルを作成します。

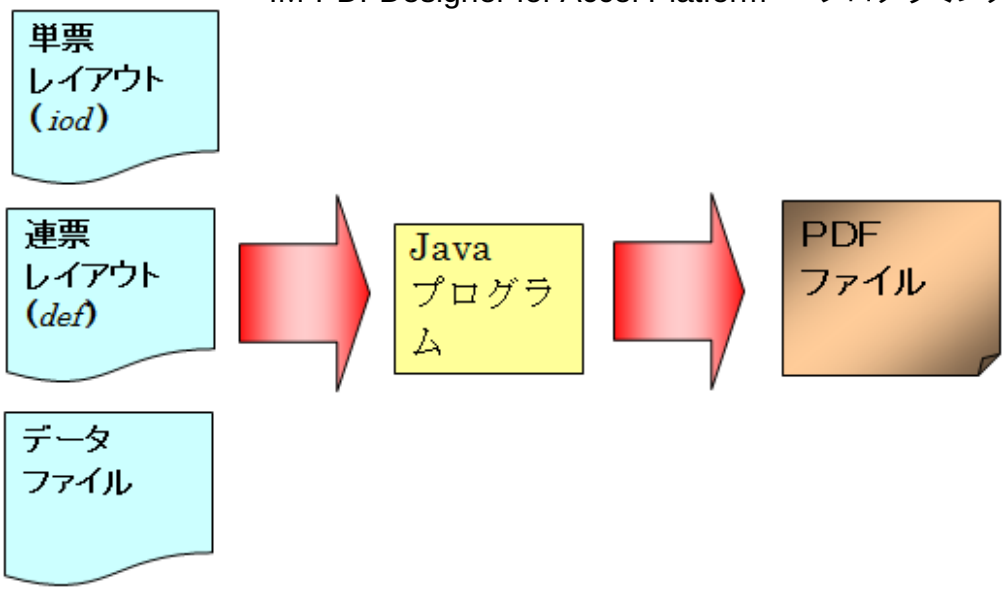


メモリデータを用いて単票形式のPDFを作成する方法については、以下のサンプルプログラムをご覧ください。

JavaEE開発モデル	jsp/pdfd/sample/docobj_act.jsp
スクリプト開発モデル	jssp/src/pdfd/sample/docobj.js

レイアウトを重ね合わせるサンプル

単票形式(IODoc用)の帳票レイアウトと連票形式(IOCela用)の帳票レイアウトを重ね合わせて、PDFファイルを作成します。
連票形式の帳票に会社ロゴを表示する等、今まで連票形式のみでは実現できなかった帳票を作成可能です。



単票レイアウトと連票レイアウトを重ね合わせてPDFファイルを作成する方法については、以下のサンプルプログラムをご覧ください。

JavaEE開発モデル	jsp/pdfd/sample/celacsvdat_act.jsp
スクリプト開発モデル	jssp/src/pdfd/sample/celacsvdat.js

サンプルプログラムの実行方法

実行方法

PDFデザイナーのサンプルプログラムは、PDFデザイナーをセットアップすることで実行することができます。

PDFデザイナーのセットアップ方法は、セットアップガイドをご覧下さい。

以下にサンプルプログラムのメニュー構成、認可・ロール構成を説明します。

メニュー構成

PDFデザイナーのサンプルメニューの構成は、以下の通りです。

PDFモジュール　サンプル		スクリプト開発モデル	単票用(IODoc用)
			連票用(IOCela用)
			PDFファイル作成(IOIntegration用)
		JavaEE開発モデル	単票用(IODoc用)
			連票用(IOCela用)
			PDFファイル作成(IOIntegration用)
	チュートリアル	スクリプト開発モデル	単票用コード(IODoc用)
			連票用コード(IOCela用)
			結合用コード(IOIntegration用)
		JavaEE開発モデル	単票用コード(IODoc用)
			連票用コード(IOCela用)
			結合用コード(IOIntegration用)

認可・ロール

PDFデザイナーのロールは、以下の通りです。

ロール名	表示名
pdfsuper	PDFデザイナー管理者

サンプルプログラムに関する注意点

本製品に付属されているサンプルプログラムは、スレッドセーフではありません。

複数のスレッドで同時にサンプルプログラムを実行した場合、正しくPDFファイルを作成できない場合があります。

各サンプルプログラムは、プログラムを理解しやすくするためにエラー処理を単純化して簡潔に記述されています。

PDFに埋め込むデータは固定値を使っています。

何回サンプルプログラムを実行しても作成されるPDFの内容は変化しません。

作成されるPDFファイルは、決められたファイル名で作成されます。

すでに同じファイル名のファイルが存在している場合は、作成されたPDFの内容で上書きしてしまいます（元のファイルは失われます）。

すべてのサンプルプログラムは画面プログラムとして作成されていますが、バッチプログラム内でも同様に（本製品で提供されている）PDF作成用APIを利用することができます。

説明を簡素化するためJSPからファイルダウンロードを行っていますが、実運用ではサーブレットを使用して下さい。

エラーコード表

エラー発生時にAPIのリターンコードとして返却されるステータスコード表です。

ステータスコード	内容
0	正常終了(エラーではありません)
-1	MS-DOSのInt21ファンクションコール4B00が無効
-2	実行ファイルが見つからない
-3	パスが見つからない
-4	ファイルオープン数エラー
-5	ダイナミックリンクライブラリ実行エラー
-6	データセグメントエラー
-7, -9	OSのメモリエラー
-8, -23～-25, -33	システムエラー
-10, -21	現在実行中のOSには未対応
-11, -20	実行に必要なファイルが壊れている
-12, -13	ランタイムのプラットフォームエラー
-14	ファイルタイプエラー
-15	実行ファイルのバージョンエラー
-16, -19	実行ファイルのロードエラー
-17	DLLのロードエラー
-18	アプリケーションのロードエラー
-22	テンポラリファイル作成失敗
-26～-32	未定義のエラー
-100	ファイルアクセスエラー
-101	パラメータエラー
-102	メモリエラー
-103	ランタイムモジュールの起動エラー
-104	IOWebDOCのセットアップエラー
-105	IOWebDOCのライセンスエラー
-106	印刷中のエラー
-107	直接印刷中のキャンセル
-200	セキュリティエラー(パスワードが不正等)

ステータスコード	内容
-999	その他のエラー
-1001	レイアウトファイルのパスが未定義
-1002	レイアウトファイルが存在しない
-1003	変換定義ファイル(cdd)のパスが未定義
-1004	変換定義ファイル(cdd)が存在しない
-1005	データファイルのパスが未定義
-1006	データファイルが存在しない
-1007	データが設定されていない
-1008	出力先PDFファイルのパスが未定義
-1009	出力先IODファイルのパスが未定義
-1010	データファイルのロードに失敗
-1011	IODOCラインタイム実行時エラー
-1012	IOWebDOC Java-Interfaceライセンスエラー
-1020	オープンパスワードとセキュリティパスワードに同じパスワードを設定している
-1021	PDFファイルのセキュリティパスワードが未設定
-1022	PDFファイルのセキュリティ情報が未設定(印刷可否、編集可否等)
その他	その他のエラー

なお、ステータスコード 0 は、リクエストされた処理を正常に終了できたことを意味しています。

したがって、ステータスコードとして 0 以外の数値が返された場合、リクエストされた処理を正常に終了できなかったことを意味します。
処理を正常終了できなかった場合は、メッセージ取得メソッドから、返却されたエラーコードに対応するエラーメッセージを取得できます (処理を正常終了している場合は、メッセージ取得メソッドから メッセージを取得する必要はありません)。

トラブルシューティング

java.lang.NoClassDefFoundErrorが発生する

クラスパスの設定が正しくありません。
セットアップガイドに従ってクラスパスを設定して下さい。

java.lang.UnsatisfiedLinkErrorが発生する

ネイティブライブラリの呼び出しができていません。
Windows Server の場合は環境変数 PATH に IOWebDOC の binフォルダのパスを設定して下さい。
Red Hat Enterprise Linux 6 の場合は、環境変数 LD_LIBRARY_PATH IOWebDOC の libディレクトリのパスを設定して下さい。

環境変数の設定については、セットアップガイドを参照して下さい。

エラーコード –1012が返される

IOWebDOC Java-Interface のライセンスが不正または有効期限切れです。
IOWebDOC Java-Interface のライセンスを正しく登録して下さい。

エラーコード –103が返される

PDF作成ランタイムに実行権限がありません。
IOWebDOC の bin ディレクトリ内のすべてのファイルに Resin を実行できるユーザの実行権限を設定して下さい。

エラーコード ー104 が返される（コンソールに「Please set up environment variable IODOC.」と表示される）

セットアップが不完全です。
セットアップガイドにしたがって環境変数を正しく設定して下さい。

エラーコード ー100が返される

ディスクがいっぱいか、ファイルにアクセスできません。
ディスク容量を確認して下さい。
また、Storage(共有ディスク)の書き込み遅延が発生している場合があるので、Storage の設定を確認して下さい。

IODoc/IOCela/IOIntegration is undefinedというエラーになる

インストールに失敗しているか、または試用期限が切れています。
正しくインストールして、ライセンスを登録して下さい。

PDFIllegalLicenseExceptionがスローされる

試用期限が過ぎています
ライセンスを登録して下さい。

iAPへのバージョンアップ時の注意事項

以下2点の作業が必要となります。
・ソースコード
・帳票レイアウト（iodファイル）

廃止メソッド

setCompressionメソッドについては、PDFデザイナー 8.x.xでは、利用できませんのでご注意下さい。

廃止クラス

Storage Service の廃止により、一部のクラスが廃止となりました。

クラス名	廃止/新規	移行先クラス
AbstractBuilder	*廃止(使用できません)*	なし
AbstractPageBuilder	*廃止(使用できません)*	なし
CompressedPDF		
IOCelaPageBuilder	*廃止(使用できません)*	CSVCela
IOCelaPageWriter	*廃止(使用できません)*	CSVCela
IODocPageBuilder	*廃止(使用できません)*	CSVDoc
IODocPageWriter	*廃止(使用できません)*	CSVDoc
PageWriter	*廃止(使用できません)*	なし
PDFBuilder	*廃止(使用できません)*	IOIntegration
PDFDocumentInformation		
PDFException		
PDFIllegalLicenseException		
PDFIllegalParameterException		
PDFIllegalStateException		
PDFIOException		
PDFLibSecurity		
PDFMemoryAccessException		
PDFRuntimeException		
PDFSecurity		
PDFUnsupportedVersionException	新規(サポート対象外のクラスを使用した場合)	
PDFWriter	*廃止(使用できません)*	IOIntegration
AbstractCSVCela	新規(インスタンスの生成はできません)	
AbstractCSVDoc	新規(インスタンスの生成はできません)	
AbstractIODOC		
AbstractIOIntegration	新規(インスタンスの生成はできません)	

クラス名	廃止/新規	移行先クラス
CSVCela		
CSVDoc		
IOIntegration		

利用可能フォント（等幅フォント）

利用フォントは、等幅フォントをご利用ください。可変幅フォントはご利用いただけません。

（例）
　MSゴシック/MS明朝 →OK 等幅フォント
　MS Pゴシック/MS P明朝 →NG 可変幅フォント

不明な場合は、個別に営業までお問合せください。

データファイルの文字コード（SJIS/MS932）

データファイルの文字コードは、SJIS（MS932）で作成してください。
データファイルを SJIS（MS932）以外で作成する必要がある場合は、個別に営業までお問合せください。

IOCELA（連票形式）の出力カスタマイズ

IOCELA（連票形式）には、設定ファイルで出力を制御する方法がございます。

影響範囲

帳票エンジンの設定ファイルに記載しますので、設定はサーバ全体に影響します。

> **コラム**
>
> 作成済みのPDFファイルは影響をうけません。

> **コラム**
>
> 設定はサーバ単位で有効になります。帳票毎に設定を切り替えることはできません。

カスタマイズ手順

1. テキストファイルを開き、INIファイル形式で「 *カスタマイズ項目* 」を記述します。
2. テキストファイルを「cela.txt」として、「%IOWEBDOC_PATH%etc」に保存します。
3. 以上で設定は完了です。

カスタマイズ項目

No.	項目	概要
1	font	半角、全角、半角カタカナ用の全てのフォントを、指定したフォントに変更します。
2	font1	半角用のフォントを、指定したフォントに変更します。
3	font2	全角用のフォントを、指定したフォントに変更します。
4	font3	半角カタカナ用のフォントを、指定したフォントに変更します。
5	mode	空白行の出力を制限します。
6	nofootspace	フッタ部の前の間隔を制御します。
7	noheadspace	ヘッダ部の後ろの間隔を制御します。
8	pattern	網掛けパターンを塗りつぶし色パターンに変換して出力します。
9	V4821compat	帳票エンジンIOWebDOC V4.8.2.1 以前と同じ方法で、パターンを出力します。

カスタマイズ項目詳細

- No.1　半角、全角、半角カタカナ用の全てのフォントを、指定したフォントに変更します。
  No.2　半角用のフォントを、指定したフォントに変更します。
  No.3　全角用のフォントを、指定したフォントに変更します。
  No.4　半角カタカナ用のフォントを、指定したフォントに変更します。

  指定例

font=フォント名
font1=フォント名
font2=フォント名
font3=フォント名

- No.5　空白行の出力を制御します。"="(イコール)の後ろに以下の何れかを指定します。

  指定例
  mode=old 空白行を出力します。
  mode=fix 空白行を出力しません。かつ、フッタ位置固定。
  mode=var 空白行を出力しません。かつ、フッタ位置可変。
  ※デフォルトは、old です。

- No.6　フッタ部の前の間隔を制御します。"="(イコール)の後ろに以下の何れかを指定します。

  指定例
  nofootspace=y フッタのスペースを空けない。
  nofootspace=n フッタのスペースを空ける。
  ※デフォルトは、n です。

- No.7　ヘッダ部の後ろの間隔を制御します。"="(イコール)の後ろに以下の何れかを指定します。

  指定例
  noheadspace=y フッタのスペースを空けない。
  noheadspace=n フッタのスペースを空ける。
  ※デフォルトは、n です。

- No.8　レイアウトファイルで指定できるパターン(1～9)に対してRGBを10進数(0～255)で指定します。

  指定例
  pattern={
  1=c 255 0 0
  2=c 0 255 0
  3=c 0 0 255
  4=c 0 255 255
  5=c 255 0 255
  6=c 255 255 0
  7=c 128 255 255
  8=c 255 128 255
  9=c 255 255 128
  }
  ※必ずc を指定してください。

- No.9　V4.8.2.1 迄は、項目のパターン（網掛け）指定の出力時に必ず罫線が出力されてしまう問題がありました。
  またV4.8.2.2 でこの問題が修正されましたが、古いレイアウトを使用した場合に、罫線が消えてしまう問題が発生する場合があります。
  このキーワードでこれを制御します。

  V4821compat V4.8.2.1 以前と同じ方法で、パターンを出力します。"="(イコール)の後ろに以下の何れかを指定します。

指定例

V4821compat=y IOWebDOC V4.8.2.1 以前と同じ方法で、パターンを出力します。

V4821compat=n IOWebDOC V4.8.2.2 以降の新しい方法で、パターンを出力します。

※デフォルトは、n です。

設定ファイル例（cela.txt）

設定ファイルのサンプルを以下に記載します。

```
#
# IOCELA runtime mode
#
#default:
# mode=old
# noheadspace=0
# nofootspace=0
# V4821compat=n
# font=
# font1=
# font2=
# font3=
#

################################################################################
# old:OldVersion, fix:FooterFixation, var:FooterVariable
#mode=old
#mode=fix
mode=var

################################################################################
# Between header block and data block
# y=no space
noheadspace=y

################################################################################
# Between header block and data block
# y=no space
nofootspace=y

################################################################################
#Since IODOC V4.8.2.2/IOWebDOC V1.8.2.2
#Pattern frame control(Pattern and no frame support)
# y=Mode is older than V4.8.2.2
V4821compat=n

################################################################################
#Since IODOC V4.8.4/IOWebDOC V1.8.4
#Font control
#font=MS 明朝
font1=Courier New
#font2=MS 明朝
#font3=MS 明朝
```

```
################################################################################
#Since IODOC V4.9.1.2/IOWebDOC V1.9.1.2
################################################################################
pattern={
1=c 255   0   0
2=c   0 255   0
3=c   0   0 255
4=c   0 255 255
5=c 255   0 255
6=c 255 255   0
7=c 128 255 255
8=c 255 128 255
9=c 255 255 128
}
```

その他（便利な機能）

帳票の実現可否の判断、帳票の実現方法がわからない、単票（IODOC）/連票（IOCela）どちらで実現すべきかの判断がつかない等の際には、営業までお問合せください。

文字サイズの自動縮小機能

入力されるデータ数が不明な場合には、文字枠を設定し、文字枠のプロパティ画面にて「文字サイズを自動縮小」にチェックをいれてください。文字数に合わせて自動的にフォントサイズを縮小する機能が利用できます。

グループ化機能の使い方

グループ化機能を利用することで、複数のオブジェクトを束ねて一つのオブジェクトとして扱うことができます。まとまった単位でのコピー、削除、移動などでご利用いただけます。

直接印刷 / FAX機能

営業までご相談ください。

ファイルサイズの縮小

同じページを大量に出力する場合（同じレイアウトが連続する帳票のケース）、以下の設定にてファイルサイズを縮小できる場合があります。

1. 該当の帳票レイアウトファイル（dlfファイル）を開きます。
2. メニューから、操作(Q)　→　ページ自動振り分け(P)　→　OKボタン　を押します。
3. 帳票レイアウトファイルを保存してください。
4. 出来上がった IODファイル をサーバ上のものと差し換えてください。
5. 以上で作業は完了です。修正前後でファイルサイズの増減をご確認ください。

> **コラム**
>
> 単票形式のみです。

> **コラム**
>
> オーバーレイページに移動したオブジェクトを編集する際には、操作(Q)　→　オーバーレイの編集　を選択してください。

柔軟な帳票レイアウト実現方法（DAT形式）

通常のレイアウトツールでは実現できないような帳票（1つの帳票の中で形式の異なる表が複数ある、非常に特殊なレイ

アウト等…）については、DAT形式で実現できる可能性があります。
DAT形式では、線や文字、識別子の内容、位置情報等をすべて外部からの命令で指定することができます。
GUIツールでレイアウトを作成するよりも実現に手間がかかりますが、レイアウトをすべて外部からの指示で指定できるため
通常では実現がむずかしいようなレイアウトも作成が可能です。
詳細は営業までお問合せください。

機能一覧（DAT形式）

機能	命令
帳票レイアウトの識別子にデータを埋め込む	識別子名 半角スペース 値
線を描画する（始点と終点の座標を指定）	LINE
線を描画する（始点の座標と長さを指定）	LINE
縦書き文字情報を出力する	PTEXT
矩形情報を出力する	RECT
横書き文字情報を出力する	RTEXT
設定値に対する倍率を指定する	UNIT
線種を指定する（線種1～5）	LTYPE
線幅を指定する（線幅1～10）	LWIDTH
フォント種別を指定する	FONT
フォントサイズを指定する	FONTSIZE
デフォルトフォントを指定する	FN
半角フォントを指定する	FN1
全角フォントを指定する	FN2
半角カタカナフォントを指定する	FNJPK
倍率を指定する（デフォルトは100%）	FWS
文字間隔を指定する	TP
全角文字の文字間隔を指定する	TP1
半角文字の文字間隔を指定する	TP2
テキストの色を指定する	TC
線の種類を指定する	LP
線の色を指定する（RGB指定）	LC
塗りつぶしの色を指定する	FC
改ページする	NEXT